K-similar Conditional Random Fields for Semi-supervised Sequence Labeling

Xi Chen[1], Shihong Chen[1,2] , Kun Xiao[1]
[1]*Computer School of Wuhan University*
[2]*National Engineering Research Center for Multimedia Software, Wuhan University*
sydchen@hotmail.com

Abstract

Sequence labeling tasks, such as named entity recognition and part of speech tagging, are the fundamental compositions of the information extraction system, and thus received attentions these years. This paper proposes k-similar conditional random fields for semi-supervised sequence labeling, and makes use of unlabeled data to calculate the similarity between words with distributional clustering. The named entity recognition experiments show that this method can improve the performance through unlabeled data.

1. Introduction

The growing availability of online textual sources and the potential number of applications of knowledge acquisition from textual data has lead to an increase in information extraction (IE) research [1]. Many of IE problems are inherently sequential in that each datum is a sequence of words (such as a sentence) rather than a single word, so algorithms can assign structure to sentences by assigning a label to each word in the sentence. These tasks are referred to as sequence labeling, such as named entity recognition and part of speech tagging. Named entity recognition (NER) has been widely researched since the 6th message understanding conference (MUC-6). The goal of NER is to recognize phrases (entities) in a document that indicate the names of persons, organizations, locations, times and quantities. It is the base step towards IE such as relation and event detection as described in automatic content extraction (ACE).

Current research on sequence labeling has focused on machine learning approaches, and one series of the methods is Markov models, including: hidden Markov models (HMMs) [2], maximum entropy Markov models (MEMMs) [3] and conditional random fields (CRFs) [4]. Linear-chain CRFs are undirected graph models in which a first-order Markov assumption is made among labels. It belongs to discriminative models, which do not make effort on the observation modeling. CRFs can make use of rich overlapped features, and solve the label bias problem in a principled way [5]. There are many extensions to CRFs, such as dynamic CRFs (DCRFs) [6], semi-Markov CRFs (semi-CRFs) [7] and skip-chain CRFs [8]. The DCRFs and skip-chain CRFs can model higher-order Markov dependencies between labels, and incorporate more features. However, the exact inferences in DCRFs and skip-chain CRFs are intractable.

Because of the limitation of the labeled data, especially when moving to the different domains, fully supervised training methods seem unreasonable. Therefore, a number of semi-supervised methods have been proposed. In addition to unlabeled data, the algorithm is provided with some supervision information, but not necessarily for all examples. Often, this information will be the targets associated with some of the examples. The interest in semi-supervised learning increased in the 1990s, mostly due to applications in natural language processing and text classification [9] [10]. Recent years, semi-supervised learning was beginning to be used in sequence labeling with Markov models. Because of the high performance of CRFs these years, some works use CRFs to semi-supervised NER [11]. In [11], a form of entropy regularization was used on the unlabeled data. It makes a precise inference under this assumption, but the training time is higher order than the supervised methods.

This paper proposes k-similar conditional random fields (k-CRF) for semi-supervised sequence labeling, and makes use of unlabeled data to calculate the similarity between words with distributional clustering [12] [13]. This method does not need additional resources such as gazetteer lists [14] to make it easy transferring to other domains. We compare the

978-0-7695-3273-8/08 $25.00 © 2008 IEEE
DOI 10.1109/ALPIT.2008.16

performance of our method when labeled and unlabeled training data is present.

2. Conditional random fields

Conditional random fields (CRFs) are undirected graphical models trained to maximize a conditional probability[4]. A common special-case graph structure is a linear chain, which corresponds to a finite state machine, and is suitable for sequence labeling. A linear-chain CRF assigns one state (label) to each observation (token), and makes a first-order Markov assumption on the states. With the observation sequence $x=\{x_1,\ldots,x_t\}$, and corresponding state sequence $y=\{y_1,\ldots,y_t\}$, a linear-chain CRF defines a conditional probability as follow:

$$P_\Lambda(y|x) = \frac{1}{Z(x)}\exp\left(\sum_{t=1}^{T}\sum_{k}\lambda_k f_k\,(y_{t-1}, y_t, x, t)\right)$$

Where the conditional probability for each state transition is represented by feature function $f_k(y_{t-1},y_t,x,t)$ associated with a learned weight λ_k. The transition in the graphical model is called a clique. $Z(x)$ is the per-input normalization that makes the probability of all state sequences sum to one:

$$Z(x) = \sum_{y}\exp(\sum_{t=1}^{T}\sum_{K}\lambda_k f_k(y_{t-1}, y_t, x, t))$$

In linear-chain CRFs, exact inference such as Viterbi labeling can be used to find the most likely label sequence y^*:

$$y^* = \mathrm{argmax}_y P_\Lambda(y|x)$$

The parameter estimation problem is to find a set of parameters Λ given training data $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$. There are typically two different ways to estimate the parameters: maximizing likelihood (ML), and Bayesian approach [15]. Commonly we use the first one to optimize the conditional log-likelihood:

$$L(\Lambda) = \sum_{i}\mathrm{log}p_\Lambda(y^{(i)}|x^{(i)})$$

Then, we use one of the limited-memory quasi-Newton methods, called L-BFGS to compute the gradient $\frac{\partial L}{\partial \lambda_k}$ and the optimized gradient $\frac{\partial p(\Lambda|D)}{\partial \lambda_k} = \frac{\partial L}{\partial \lambda_k} - \frac{\partial \lambda_k}{\sigma^2}$. L-BFGS has previously been shown to outperform other optimization algorithms for linear-chain CRFs [16].

3. Semi-supervised sequence labeling

In semi-supervised learning, one of the most important problems is how to use the unlabeled data. In this section, we will introduce the k-similar conditional random fields to perform this task.

3.1. K-similar CRFs

When labeling a word not seen before, one usually labels it using similar cases. For example, "Bill Gates" was labeled as a person name in training data, and without any reference of "Bill Clinton". But "Bill Clinton" appears in the unlabeled data. And "Clinton" is distributional similar to "Gates", as shown in table 1. Therefore, the word "Clinton" is most likely labeled as a person name too. With this intuition, we propose k-similar CRFs to perform semi-supervised sequence labeling. Our proposed method relies heavily on the assumption that the unseen word can be labeled using the most frequent labels of similar words. The k-CRFs can incorporate the similarity features into linear-chain CRFs, and take advantages of both the k-nearest neighbor algorithm in unseen words labeling and CRFs in sequence labeling.

A k-CRF defines a conditional probability as follow:

$$P_\Lambda(y|x) = \frac{1}{Z(x)}\exp\left(\sum_{x'\in S(x)}(\theta\cdot\Lambda)^T f(y_{i-1}, y_i, x', i)\right)$$

Where $S(x) = \{x^{(1)},\ldots,x^{(k)}\}$ is the set of k most similar words of x which appears in the training data, and $\theta = \{\theta^{(1)},\ldots,\theta^{(k)}\}$ is the corresponding normalized similarity vector that satisfies $\sum_{i=1}^{k}\theta^{(i)} = 1$. f is the feature vector with features for $S(x)$, and Λ is the corresponding weight vector. $\theta\cdot\Lambda$ denotes the final weight vector when word x is transferred to its similar words. The value of k can be determined in the experiment. $Z(x)$ is the per-input normalization that makes the probability of all state sequences sum to one:

$$Z(x) = \sum_{y}\exp\left(\sum_{x'\in S(x)}(\theta\cdot\Lambda)^T f(y_{i-1}, y_i, x', i)\right)$$

3.1.1. Inference. The inference problem is, given an input string x, to compute the most likely labeling $\mathrm{argmax}_y P_\Lambda(y|x)$. In k-CRFs, given the parameter

vector Λ, the best labeling for a sequence can also be found exactly using the Viterbi algorithm.

The algorithm maintains the un-normalized probability of the best labeling ending at position i with the label y for each tuple of the form (i, y). The labeling itself is also stored along with the probability. Denoting the best un-normalized probability for (i, y) by V(i, y), we can calculate V(i,y) by the following recurrence:

$$V(i,y) = \max_{y'}\left(V(i-1,y') \cdot \exp\left(\sum_{x' \in S(x)} (\theta \cdot \Lambda)^T f(y,y',x',i)\right)\right)$$

The base case is:

$$V(0,y) = [\![y = 'start']\!]$$

$[\![y = 'start']\!]$ denotes whether the variable y equals 'start', that is the start of a sequence. The normalized probability of the best labeling is given by $\frac{\max_y V(n,y)}{Z_x}$ and the labeling itself is given by $\text{argmax}_y V(n,y)$.

3.1.2. Training. To prevent over-fitting, the optimized conditional log-likelihood of a set of training instances (x^k, y^k) using parameters Λ is given by:

$$L(\Lambda) = \sum_k \left((\theta \cdot \Lambda)^T \cdot F(y^k, x^k) - \log Z_\Lambda(x^k)\right) - \frac{||\Lambda||^2}{2\sigma^2}$$

The gradient of the log-likelihood is given by:

$$\nabla L(\Lambda) = \sum_k \left(\theta^T F(y^k, x^k) - E_{P(y|x^k)}[\theta^T F(y, x^k)]\right) - \frac{\Lambda}{\sigma^2}$$

The j^{th} entry of $F(y^k, x^k)$ is given by:

$$F_j(y^k, x^k) = \sum_i \sum_{x^{k'} \in S(x^k)} f_j(y_i, y_{i-1}, x^{k'}, i)$$

And $E_{P(y|x^k)}[\theta^T F(y, x^k)]$ is the expected value of the global feature vector under the conditional probability distribution. The j^{th} entry of $F(y, x^k)$ is given by:

$$F_j(y, x^k) = \sum_i \sum_{x^{k'} \in S(x^k)} f_j(y_i, y_{i-1}, x^{k'}, i)$$

Therefore, we can calculate $E_{P(y|x^k)}[\theta_j^T F_j(y, x^k)]$ using forward and backward algorithm:

$$E_{P(y|x^k)}[\theta_j^T F_j(y, x^k)] = \sum_i E_{P(y|x^k)}\left(\sum_{x^{k'} \in S(x^k)} \theta_j^T f_j(y_i, y_{i-1}, x^{k'}, i)\right) = \sum_i \sum_{y',y} \alpha(y', i-1) \cdot Q_i(y', y) \cdot \beta(y, i)$$

Where α and β are called the forward and backward vectors. $\alpha(y, i)$ denotes the un-normalized probability of a partial labeling, ending at position i with label y. Similarly, $\beta(y, i)$ denotes the un-normalized probability of a partial segmentation starting at position i + 1 assuming a label y at position i. α and β and can be computed via the following recurrences:

$$\alpha(y,i) = \sum_{y'} \alpha(y', i-1) \cdot \exp\left(\sum_{x^{k'} \in S(x^k)} (\theta \cdot \Lambda)^T f(y, y', x^{k'}, i)\right)$$

$$\beta(y,i) = \sum_{y'} \beta(y', i+1) \cdot \exp\left(\sum_{x^{k'} \in S(x^k)} (\theta \cdot \Lambda)^T f(y', y, x^{k'}, i+1)\right)$$

The base cases are:

$$\alpha(y, 0) = [\![y = 'start']\!]$$
$$\beta(y, n+1) = [\![y = 'stop']\!]$$

and Q_i is a matrix s.t.

$$Q_i(y', y) = \sum_{x^{k'} \in S(x^k)} \theta_j^T f_j(y, y', x^{k'}, i) \cdot \exp\left(\sum_{x^{k'} \in S(x^k)} (\theta \cdot \Lambda)^T f(y, y', x^{k'}, i))\right)$$

Thus, after all the α, β vectors and Q matrices have been computed, the gradient can be easily obtained. And we can use the L-BFGS to maximize the log-likelihood.

3.2. Distributional similarity

Distributional similarity[12] is a well known method for grouping words based on local context. The working hypothesis is that syntactic behavior is reflected in co-occurrence patterns. Therefore, the syntactic behavior of a word is with respect to its left and right context. And the words with the similar syntactic behavior share the similar contexts. It is the only way that limited prior information in the form of a prototype list could be helpful in tagging non-prototype words. Our method can use these relationships between words.

3.2.1. Context representation. For each unique word, we can use a number of words to represent its context. Typically, the words with low frequency can not reflect word similarity accurately because of the high error probability. And large number of context dimension will result in large similarity matrix in the singular value decomposition (SVD), increasing the time and space complexity. As a matter of fact, collecting a context vector of a fixed number of the most frequently occurring words conjoined with a direction and distance will be enough. With these words occurring in the nearby positions, it will form a |D| dimension context vector for each word. And it totally makes a |V|×|D| context matrix A for all words.

3.2.2. Word similarity. There are four commonly used association coefficients suitable for measuring the textual data similarity, namely, the inner product, the cosine coefficient, the Dice coefficient and the Jaccard coefficient. The cosine coefficient measures the similarity between two objects by calculating the cosine of the angle between their corresponding vectors in the vector space. And it is sensitive to the relative importance of features, such as which is the strongest feature, and which is the weakest. This characteristic has made the cosine coefficient popular for comparing pairs of words in terms of their contextual similarity.

It is not working well when using cosine measure directly because of data sparseness. Consider two infrequent adjectives that happen to modify different nouns in the corpus. Their right similarity according to the cosine measure would be zero. This is clearly undesirable. The similarity of the two nouns should spread across the corpus, and increase the similarity of the two adjectives. One solution to this problem is the application of singular value decomposition. Find the singular value decomposition $A = U\Sigma^{T}V$ and the rows of U is the word vector for calculating distributional similarity.

4. Experiments

In our experiments, the training and testing corpus is from CoNLL2004. The data consist of the sections of the Wall Street Journal in 1990s, part of the Penn Treebank. It has 5516 sentences and 272759 words. And we use Reuters-21578 corpus and Usenet newsgroups collection for unlabeled data. The two corpuses are collection of news documents, and have totally 9.4 million words. We use no additional linguistic resources for our experiments.

To avoid over-fitting, some pre-process must be carried out, including number recognition, year recognition, and etc. The sentences of the unlabeled corpus should be recognized from the passages. The words and punctuations should also be recognized from sentences. Word stemming is not necessary here because the word form contain useful information for sequence labeling.

We only use the lower case of words to calculate their similarity. In this experiment, we choose 400 most frequently occurring words in both labeled and unlabeled corpuses. For all words in the training and testing data, calculate their context vector, including two positions before and after the words. This will produce a 1600 dimensional context vector for each word, and totally a 14595×1600 context matrix A for all words. The singular value decomposition of matrix A is performed to avoid the data sparseness. The examples of the most similar words in training data are shown in the Table 1. It can be improved by extending the unlabeled corpus from internet or increasing the dimension of context vector. This will also make the final experiment result more accurate.

Table 1. Examples of the most similar words

six-year	15-day grace break-in year-earlier waiting 5-year statement 30-day three-year second
fla.	fla. ariz. colo. ala. n.c. mich. s.c. nev. ore. ky.
birthplace	avenue fields beach omaha seattle boise hometown street mayor mirror
clinton	clinton clare gates higgins directs authorizes carlson keene barber barnes
lab	lab showcase scientist institute triangle parkway coordinator laboratory center rebels

In our named entity recognition experiments with CoNLL2004, there are four types of named entities: 4886 persons (PER), 6302 locations (LOC), 5644 organizations (ORG) and 6614 miscellaneous (MISC). We use only word features and word form features such as initials and affix in both methods. And a threshold 0.7 is used to discard the dissimilar words. These words may affect the training and testing when they appear frequently.

To make a significant evaluation on our method, we conduct two experiments, one using randomly chose 10% labeled data for training and the rest 90% for testing, the other using 90% for training and the rest 10% for testing. The first experiment stands for the circumstance with poor training data, the second stands for rich training data circumstance. In both experiments, we compare the effect of the parameter k for named entity recognition. In these experiments, we evaluate our system in terms of precision (P), recall (R) and F1-measure. The overall precision, recall and F1-score using different parameter k are shown in Table 2.

Table 2. Performance of k-CRFs using different parameter k

K	Training data: 10%			Training data: 90%		
	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
1	65.33	42.93	51.81	71.98	47.70	57.38
2	68.42	46.16	55.13	76.21	56.18	64.68
3	68.66	**48.08**	**56.56**	76.02	**58.87**	**66.35**
4	68.69	47.82	56.38	76.16	58.31	66.05
5	69.25	46.92	55.94	76.48	56.37	64.90
6	69.45	42.99	53.11	75.67	54.91	63.64
7	65.25	45.91	53.89	77.23	55.58	64.64
8	67.23	44.78	53.76	**80.83**	46.24	58.83
9	**70.31**	42.03	52.61	76.82	53.80	63.28
10	68.63	42.68	52.63	76.73	54.55	63.77

Table 3. Comparing results for k-CRF and linear-chain CRF

Training data: 10%						
Entity types	k-CRF			Linear-chain CRF		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
ORG	**59.83**	33.07	**42.60**	55.38	**34.18**	42.27
LOC	**67.38**	**44.75**	**53.78**	63.97	42.68	51.20
MISC	64.43	**54.02**	**58.77**	**67.74**	41.85	51.74
PER	**78.90**	**57.27**	**66.37**	63.75	51.24	56.81
Overall	**68.66**	**48.08**	**56.56**	63.10	43.12	51.24

Training data: 90%						
Entity types	k-CRF			Linear-chain CRF		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
ORG	**71.93**	**50.09**	**59.06**	59.15	46.93	52.34
LOC	73.63	**58.27**	**65.06**	**76.94**	42.77	54.98
MISC	**73.23**	52.55	61.19	72.85	**52.87**	**61.27**
PER	**82.57**	**71.51**	**76.64**	71.43	52.89	60.78
Overall	**76.02**	**58.87**	**66.35**	69.99	48.93	57.60

We get the best F1-score when the parameter k=3 in both experiments. The comparing results for our method using k-CRFs and the original method using linear-chain CRFs are shown in the Table 3. The precision and recall of k-CRF is higher in most kind of entities. The overall precision, recall and F1 are increased by 5.56%, 4.96% and 5.32% to the linear-chain CRFs in experiment using 10% labeled data for training, and 6.03%, 9.94% and 8.75% in experiment using 90% labeled data for training. And it can be improved through a larger and domain related unlabeled data.

We have also compared our method to the state-of-art semi-supervised learning algorithm, self-learning[17] or bootstrapping, in natural language processing. However, the performance of self-learning was even worse than the standard supervised linear-chain CRFs.

5. Conclusions

K-similar CRF is an efficient extension of CRF that incorporate the similarity features into linear-chain CRFs, and take advantages of both k-nearest neighbor algorithm in unseen words labeling and CRFs in sequence labeling. This method relies on the assumption that the unseen word can be labeled using the most frequent labels of similar words, and makes use of unlabeled data to calculate the similarity between words with distributional clustering. The training complexity of k-CRF is the same order as the linear-chain CRF. The experiments show that our method outperforms the standard supervised CRF method.

While our system has achieved a promising performance, there is still much to be done to improve it. First, the similar words calculated from unlabeled corpus are not accurate enough because of the limitation of data and calculating time. Second, the granularity of the similar words can be extended to similar phrase with semi-Markov assumption. For future work, we intend to explore more effective word and phrase similarity algorithm to further improve the results of k-CRFs.

6. References

[1] Turmo, J., A. Ageno, and N. Catala, Adaptive information extraction. Acm Computing Surveys, 2006. 38(2).

[2] G.D., Z. and S. J. Named entity recognition using an HMM-based chunk tagger. in Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL) 2002.

[3] McCallum, A., D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. in International Conference on Machine Learning (ICML). 2000.

[4] Lafferty, J., A. McCallum, and F. Pereira, Conditional random fields: Probabilistic models for segmenting and

labeling sequence data, in International Conference on Machine Learning. 2001.

[5] Nguyen, N. and Y. Guo. Comparisons of sequence labeling algorithms and extensions. in Proceedings of the 24th international conference on machine learning (ICML). 2007.

[6] Sutton, C., A. McCallum, and K. Rohanimanesh, Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. Journal of Machine Learning Research, 2007. 8, pp. 693-723.

[7] Sarawagi, S. and W. Cohen. Semi-Markov conditional random fields for information extraction. in International Conference on Machine Learning (ICML). 2004.

[8] Sutton, C. and A. McCallum, Collective segmentation and labeling of distant entities in information extraction, in ICML workshop on statistical relational learning and its connections to other fields. 2004.

[9] Nigam, K., et al. Learning to classify text from labeled and unlabeled documents. in Proc. of the 15th national conference on artificial intelligence. 1998.

[10] Blum, A. and T. Mitchell. Combining labeled and unlabeled data with co-training. in Proc. of the 11th annual conference on computational learning theory. 1998.

[11] F., J., et al. Semi-supervised conditional random fields for improved sequence segmentation and labeling. in Proc. of the 21st international conference on computational linguistics. 2006.

[12] Schutze, H. Distributional part-of-speech tagging. in Proc. of the European chapter of the association for computational linguistics (EACL). 1995.

[13] Freitag, D. Trained named entity recognition using distributional clusters. in Proc. of Empirical Methods on Natural Language Processing (EMNLP). 2004.

[14] Kozareva, Z., bootstrapping named entity recognition with automatically generated gazetteer lists, in Association for computational linguistics. 2006.

[15] Qi, Y., M. Szummer, and T.P. Minka. Bayesian Conditional Random Fields. in Proc. of the Tenth Internationol Workshop on Artijicial Intelligence and Statistics. 2005.

[16] Malouf, R. A comparison of algorithms for maximum entropy parameter estimation. in Conference on Natural Language Learning (CoNLL). 2002.

[17] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. in Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics, 1995. pp. 189-196